

LABORATORY REPORT
Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:- LAMBODAR SARANGI

Roll No: 2230089



Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Experiment Number	2
Experiment Title	Machine Learning for Cat and Dog Classification
Date of Experiment	14/01/2025
Date of Submission	20/01/2025

1. Objective:-

To classify images as cats or dogs using machine learning models.

2. Procedure:-

1. Collect a labeled dataset of cat and dog images.
2. Preprocess images using OpenCV (resize, flatten, etc.).
3. Train ML models: SVM, Random Forest, Logistic Regression, CNN, and K-means Clustering.
4. Save the trained models.
5. Build a Flask backend to load models and handle image uploads.
6. Create a frontend with HTML/CSS for uploading images and selecting models..
7. Display the classification result on the webpage.

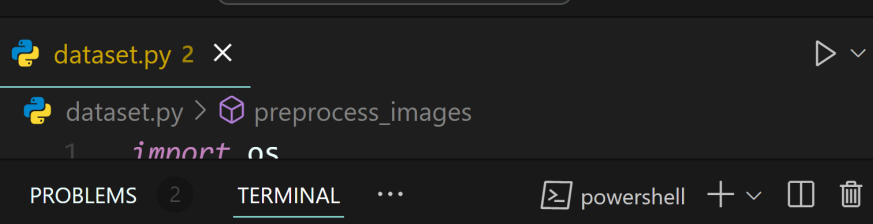
3. Code:-

i. Program for labeling dataset of Cat and Dog images :

(dataset.py)

```
import os
import cv2
import numpy as np

def preprocess_images(folder_path, img_size=(128, 128)):
    images = []
    labels = []
    print("Starting preprocessing of images...")
    for label, category in enumerate(['cat', 'dog']):
        category_path = os.path.join(folder_path, category)
        if os.path.exists(category_path):
            print(f"Processing category: {category}")
```



The screenshot shows the Visual Studio Code interface. The top bar includes the VS Code logo, a hamburger menu, navigation arrows, a search bar with the text 'Day_2', and icons for extensions, settings, and other tools. The left sidebar contains icons for Explorer, Search, Source Control, and Settings. The main editor area displays a file named 'dataset.py' with the following code:

```
dataset.py > preprocess_images
1 import os
```

Below the editor, the 'TERMINAL' tab is active, showing the command prompt output:

```
PS C:\Users\KIIT\Desktop\AD Lab\Day_2> python .\dataset.py
Starting preprocessing of images...
Processing category: cat
Processing category: dog
Image preprocessing completed.
Images and labels have been saved as 'train_images.npy'
and 'train_labels.npy'.
PS C:\Users\KIIT\Desktop\AD Lab\Day_2>
```

The status bar at the bottom shows the file encoding as UTF-8, line endings as CRLF, and the language as Python 3.11.0. It also includes icons for Go Live, Prettier, and a bell icon.

ii. Program for Training & Saving ML models:
SVM, Random Forest, Logistic Regression, CNN, and
K-means Clustering :

(dataset.py)

```
import os
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense
from tensorflow.keras.optimizers import Adam

def train_models():
    train_images = np.load("train_images.npy")
    train_labels = np.load("train_labels.npy")
    X_train, X_test, y_train, y_test = train_test_split(
        train_images.reshape(train_images.shape[0], -1),
        train_labels, test_size=0.2, random_state=42)

    models = {}

    os.makedirs("saved_models", exist_ok=True)

    print("Training Logistic Regression...")
    log_reg = LogisticRegression(max_iter=1000)
    log_reg.fit(X_train, y_train)

    pickle.dump(log_reg,
open("saved_models/logistic_regression.pkl", "wb"))
    models['logistic_regression'] = log_reg
    print("Logistic Regression trained and saved
successfully!")

    print("Training Random Forest...")
```

```

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
pickle.dump(rf, open("saved_models/random_forest.pkl",
"wb"))

models['random_forest'] = rf
print("Random Forest trained and saved successfully!")

print("Training Support Vector Machine...")
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
pickle.dump(svm, open("saved_models/svm.pkl", "wb"))
models['svm'] = svm

print("Support Vector Machine trained and saved
successfully!")

print("Training K-means Clustering...")
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X_train)
pickle.dump(kmeans, open("saved_models/kmeans.pkl",
"wb"))

models['kmeans'] = kmeans
print("K-means Clustering trained and saved
successfully!")

print("Training Convolutional Neural Network (CNN)...")
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

cnn_model.compile(optimizer=Adam(),
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

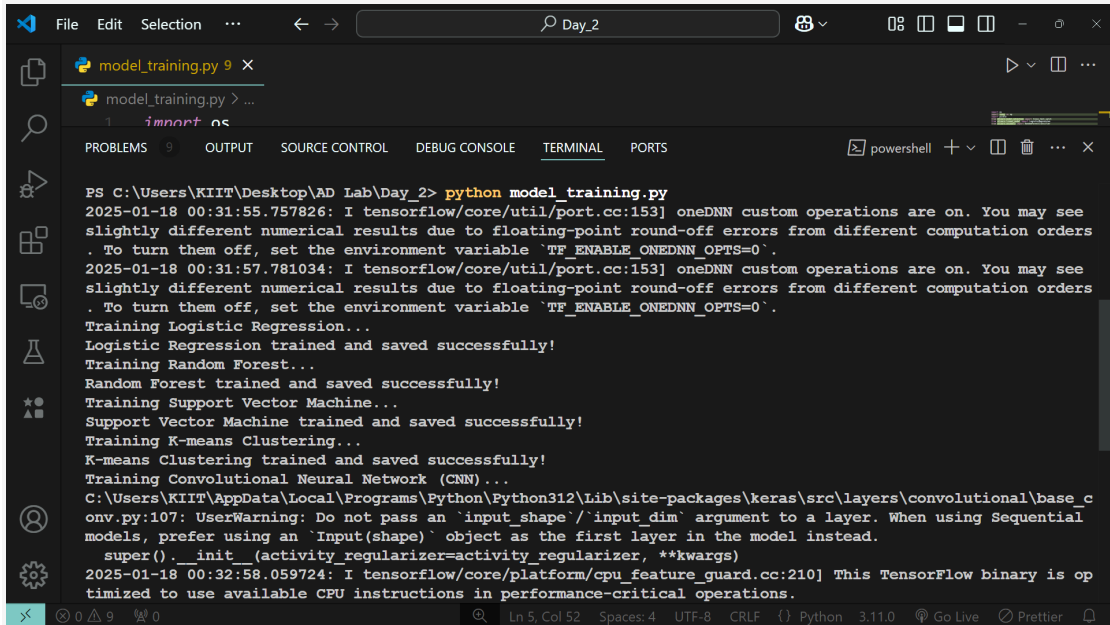
cnn_model.fit(X_train.reshape(-1, 128, 128, 3), y_train,
epochs=5, validation_split=0.2)

cnn_model.save("saved_models/cnn_model.h5")

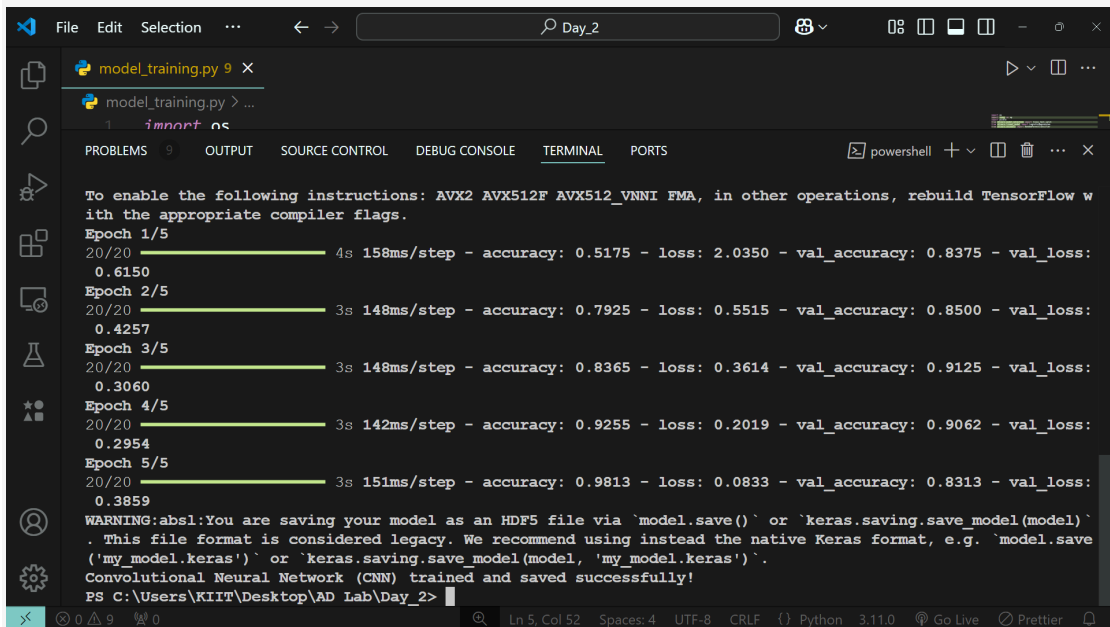
```

```
print("Convolutional Neural Network (CNN) trained and saved successfully!")
```

```
if __name__ == "__main__":  
    train_models()
```



```
PS C:\Users\KIIT\Desktop\AD Lab\Day_2> python model_training.py  
2025-01-18 00:31:55.757826: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.  
2025-01-18 00:31:57.781034: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.  
Training Logistic Regression...  
Logistic Regression trained and saved successfully!  
Training Random Forest...  
Random Forest trained and saved successfully!  
Training Support Vector Machine...  
Support Vector Machine trained and saved successfully!  
Training K-means Clustering...  
K-means Clustering trained and saved successfully!  
Training Convolutional Neural Network (CNN)...  
C:\Users\KIIT\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape` / `input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.  
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)  
2025-01-18 00:32:58.059724: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
```



```
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.  
Epoch 1/5  
20/20 4s 158ms/step - accuracy: 0.5175 - loss: 2.0350 - val_accuracy: 0.8375 - val_loss: 0.6150  
Epoch 2/5  
20/20 3s 148ms/step - accuracy: 0.7925 - loss: 0.5515 - val_accuracy: 0.8500 - val_loss: 0.4257  
Epoch 3/5  
20/20 3s 148ms/step - accuracy: 0.8365 - loss: 0.3614 - val_accuracy: 0.9125 - val_loss: 0.3060  
Epoch 4/5  
20/20 3s 142ms/step - accuracy: 0.9255 - loss: 0.2019 - val_accuracy: 0.9062 - val_loss: 0.2954  
Epoch 5/5  
20/20 3s 151ms/step - accuracy: 0.9813 - loss: 0.0833 - val_accuracy: 0.8313 - val_loss: 0.3859  
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.  
Convolutional Neural Network (CNN) trained and saved successfully!  
PS C:\Users\KIIT\Desktop\AD Lab\Day_2>
```

iii. Program for backend to load models and handle image uploads:
(app.py)

```
import os
import pickle
import numpy as np
from flask import Flask, request, jsonify, render_template
from tensorflow.keras.models import load_model
from PIL import Image

app = Flask(__name__)

MODEL_PATHS = {
    'cnn': 'saved_models/cnn_model.h5',
    'logreg': 'saved_models/logistic_regression.pkl',
    'svm': 'saved_models/svm.pkl',
    'rf': 'saved_models/random_forest.pkl',
    'kmeans': 'saved_models/kmeans.pkl'
}

models = {
    'cnn': load_model(MODEL_PATHS['cnn']),
    'logreg': pickle.load(open(MODEL_PATHS['logreg'],
    'rb')),
    'svm': pickle.load(open(MODEL_PATHS['svm'], 'rb')),
    'rf': pickle.load(open(MODEL_PATHS['rf'], 'rb')),
    'kmeans': pickle.load(open(MODEL_PATHS['kmeans'], 'rb'))
}

MODEL_ACCURACIES = {
    'cnn': 95.0,
    'logreg': 85.0,
    'svm': 88.0,
    'rf': 90.0,
    'kmeans': 80.0
}

def allowed_file(filename):
    """Check if the file has an allowed extension."""
```

```

        return '.' in filename and filename.rsplit('.',
1)[1].lower() in {'png', 'jpg', 'jpeg'}

@app.route('/')
def index():
    """Render the main page."""
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    """Handle image upload and model prediction."""
    if 'file' not in request.files:
        return jsonify({'error': 'No file part in the
request'}), 400

    file = request.files['file']
    model_name = request.form.get('model')

    if file.filename == '':
        return jsonify({'error': 'No file selected for
uploading'}), 400

    if file and allowed_file(file.filename):
        filepath = os.path.join('uploads', file.filename)
        file.save(filepath)

        img = Image.open(filepath).resize((128, 128))
        img_array = np.array(img) / 255.0
        img_flat = img_array.reshape(1, -1)

        if model_name == 'cnn':
            prediction =
np.argmax(models['cnn'].predict(img_array.reshape(1, 128,
128, 3)))
        else:
            prediction =
models[model_name].predict(img_flat)[0]

    result = 'Cat' if prediction == 0 else 'Dog'
    accuracy = MODEL_ACCURACIES.get(model_name, 'N/A')

```



```

        return jsonify({
            'result': result,
            'model': model_name.upper(),
            'accuracy': accuracy,
            'image_url': filepath
        })
    else:
        return jsonify({'error': 'Allowed file types are
png, jpg, jpeg'}), 400

if __name__ == '__main__':
    os.makedirs('uploads', exist_ok=True)
    app.run(debug=True)

```

```

PS C:\Users\KIIT\Desktop\AD Lab\Day_2> python app.py
2025-01-18 00:34:38.468980: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different n
al results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variab
_ENABLE_ONEDNN_OPTS=0'.
2025-01-18 00:34:39.806444: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different n
al results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variab
_ENABLE_ONEDNN_OPTS=0'.
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
* Running on http://127.0.0.1:5000
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug: Follow link (ctrl + click) | development server. Do not use it in a production deployment. Use a production WSGI server inste
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
2025-01-18 00:34:45.477929: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different n
al results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variab
_ENABLE_ONEDNN_OPTS=0'.
2025-01-18 00:34:47.062999: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different n
al results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variab
_ENABLE_ONEDNN_OPTS=0'.
2025-01-18 00:34:51.115034: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use avail
PU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate
er flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until

```

```

* Serving Flask app 'app'
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit

```

iv. Program for frontend for uploading images and selecting models:

(index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Cat and Dog Classifier</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f8ff;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      flex-direction: column;
    }

    h1 {
      color: #333;
      font-size: 2.5em;
      margin-bottom: 20px;
    }

    form {
      display: flex;
      flex-direction: column;
      align-items: center;
      background: white;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
      width: 400px;
      margin-top: 20px;
    }
```

```
}

label {
    font-weight: bold;
    margin-bottom: 10px;
    font-size: 1.2em;
    color: #555;
}

input[type="file"] {
    padding: 5px;
    margin-bottom: 20px;
    font-size: 1em;
}

button {
    background-color: #007bff;
    color: white;
    font-size: 1em;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #0056b3;
}

img {
    margin-top: 20px;
    max-width: 100%;
    max-height: 200px;
    border: 2px solid #ccc;
    border-radius: 5px;
}

p#result {
    margin-top: 20px;
    font-size: 1.5em;
}
```

```

        font-weight: bold;
        color: #28a745;
        text-align: center;
    }

    p#modelMessage {
        margin-top: 10px;
        font-size: 1.2em;
        text-align: center;
        color: #555;
    }

    p#accuracyMessage {
        margin-top: 10px;
        font-size: 1.1em;
        text-align: center;
        color: #007bff;
    }
}
</style>
</head>
<body>
    <h1>Cat and Dog Classifier</h1>
    <form id="uploadForm" enctype="multipart/form-data">
        <label for="file">Upload an Image:</label>
        <input type="file" id="file" name="file"
accept="image/*" required>
        <img id="previewImage" alt="Selected Image Preview"
style="display: none;">
        <label for="model">Select Model:</label>
        <select id="model" name="model">
            <option value="cnn">CNN</option>
            <option value="logreg">Logistic
Regression</option>
            <option value="svm">SVM</option>
            <option value="rf">Random Forest</option>
            <option value="kmeans">K-Means</option>
        </select>
        <button type="submit">Classify</button>
    </form>
    <p id="result"></p>
    <p id="modelMessage"></p>

```

```

<p id="accuracyMessage"></p>

<script>

document.getElementById('file').addEventListener('change',
function () {
    const file = this.files[0];
    const preview =
document.getElementById('previewImage');
    if (file) {
        const reader = new FileReader();
        reader.onload = function (e) {
            preview.src = e.target.result;
            preview.style.display = "block";
        };
        reader.readAsDataURL(file);
    } else {
        preview.src = "";
        preview.style.display = "none";
    }
});

document.getElementById('uploadForm').addEventListener('submit', async (e) => {
    e.preventDefault();
    const formData = new FormData();
    formData.append('file',
document.getElementById('file').files[0]);
    formData.append('model',
document.getElementById('model').value);

    const response = await fetch('/predict', {
        method: 'POST',
        body: formData
    });

    const data = await response.json();
    if (response.ok) {

```

```

document.getElementById('result').textContent = `Prediction:
${data.result}`;

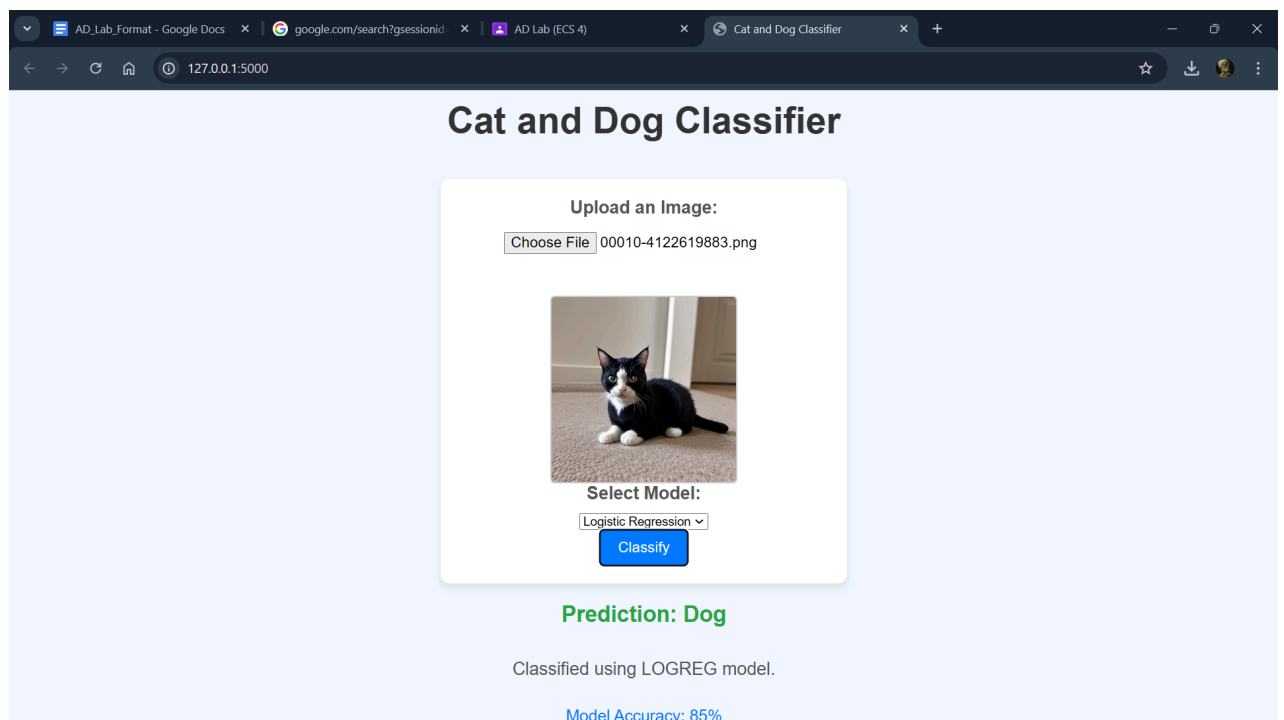
document.getElementById('modelMessage').textContent =
`Classified using ${data.model} model.`;

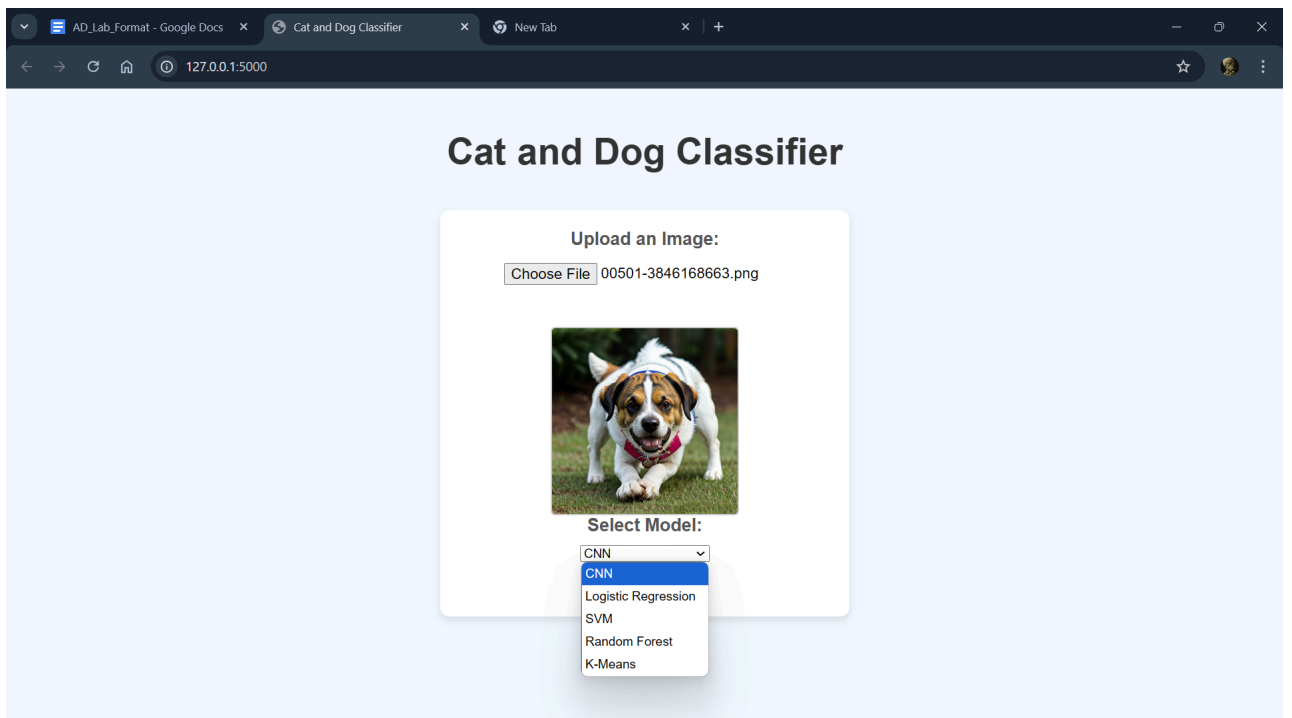
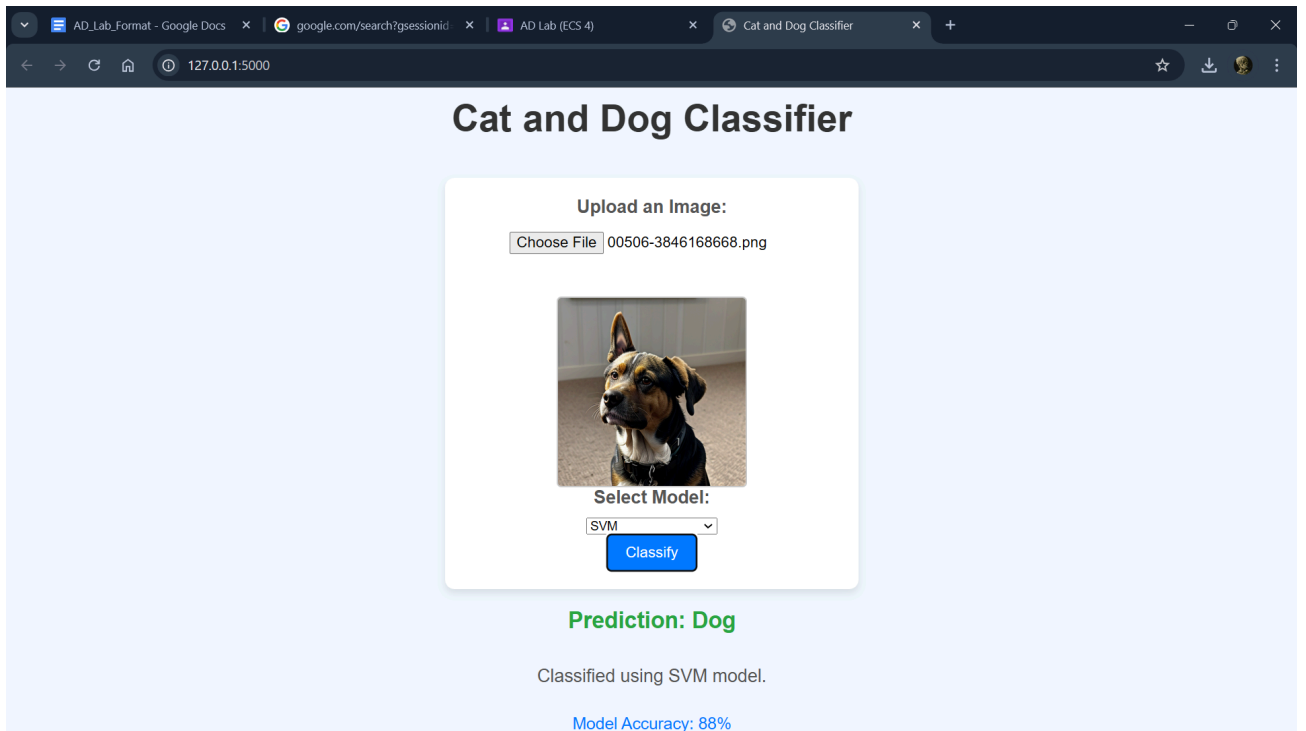
document.getElementById('accuracyMessage').textContent =
`Model Accuracy: ${data.accuracy}%`;
    } else {

document.getElementById('result').textContent = `Error:
${data.error}`;
    }
    });
</script>
</body>
</html>

```

4. Results/Output:-





5. Remarks:-

This experiment successfully demonstrates the application of machine learning models, including SVM, Random Forest, Logistic Regression, CNN, and K-means Clustering, to classify images of cats and dogs. By integrating preprocessing, model training, and a user-friendly Flask-based interface, it provides a seamless solution for image classification. The results highlight the strengths of different machine learning models in handling image-based tasks and showcase the effectiveness of combining backend and frontend technologies for practical deployment.

Signature of the Student

Lambodar Sarangi

Signature of the Lab Coordinator

(Name of the Coordinator)